

thesis/  
reports  
Campbell,  
G. S.

Simulation of Soil Heating Under  
Smoldering Duff Fires

Washington State University

FS Contact: James K. Brown

Co-Op Contact: Gaylon S. Campbell

## **Simulation of Soil Heating Under Smoldering Duff Fires**

**Gaylon S. Campbell  
Dept. of Crop and Soil Sciences  
Washington State University  
Pullman, WA 99164-6420**

**Prepared for the  
Intermountain Fire Sciences Laboratory  
U. S. Forest Service  
Missoula, MT**

## Introduction

Smoldering duff fires involve the pyrolysis and oxidation of tightly packed fuel materials on the forest floor. Spread rates of duff fires are slow; three orders of magnitude slower than flaming (Hungerford et al. 1991). Fire temperatures are lower than for flaming, but heating of deep soil layers by the fire is often greater because of the intimate contact of the fire with the mineral surface and the long duration of the fire.

Effects of fire can be beneficial or detrimental, depending on the duration and intensity of the fire and the initial condition of the soil. The purpose of this report is to present results of simulations which are intended to quantify effects of fire intensity and duration and soil moisture conditions on soil heating under smoldering duff fires. The model used simulations linked transport of heat and water in soil. It was developed under a previous agreement (Campbell et al. 1992). That model was tested by comparing its results with measurements made on soil columns in the laboratory. The columns were heated at the surface by a propane burner. The radiant heat from the burner was measured, and this measurement was used as one boundary condition for the model. Excellent agreement was obtained between modeled and measured temperature profiles in the soil.

## Assumptions for the Duff Fire Model

Fire in dry duff spreads horizontally, with a parabolic front, at a rate of about 3 cm/hr (Hungerford et al., 1991). To completely model this phenomenon would require a two or three dimensional soil heat flow model as well as an appropriate model for heat production, spread rate, and heat partitioning. The differential equations governing linked transport of heat and water in soil are highly nonlinear. One-dimensional simulations are a challenge, and multidimensional simulations are even more challenging. We therefore followed Steward et al. (1990) and Pafford et al. (1991) in assuming that temperature gradients in the horizontal are small compared to those in the vertical. We therefore assumed we could approximate heat flow into the soil using our one-dimensional model.

By knowing the depth and density of duff, we can compute the total amount of heat which is released when it burns ( $\text{J/m}^2$ ):

$$Q_o = \rho_b d Q_e \quad (1)$$

where  $\rho_b$  is the bulk density of the duff ( $\text{kg/m}^3$ ),  $d$  is the duff depth (m), and  $Q_e$  is the heat of combustion of the duff ( $\text{J/kg}$ ). Frandsen (1991a) gives a figure of 14.2 MJ/kg for  $Q_e$  of dry peat moss with negligible mineral content. The Handbook of Chemistry and Physics gives the heat of combustion of pine wood as 18.4 MJ/kg. This heat is released over a period of time determined by the spread rate of the fire. Frandsen (1991b) presents methods for estimating spread rate, but we were not able to incorporate those into the present model. Instead we assumed the rate of heat release to follow a Gaussian distribution (Steward et al. 1990):

$$q = q_m \exp \left[ - \left( \frac{t - t_m}{w} \right)^2 \right] \quad (2)$$

where  $t$  is time,  $t_m$  is the time at maximum heat input,  $w$  is an index of burn time (68% of the heat is released between the time  $t = -w$  and  $t = w$ ) and  $q_m$  is the maximum rate of heat input ( $\text{W/m}^2$ ). The integral of  $q$  over the entire burn time is equal to the total heat input to the surface, which is some fraction of the heat released by burning the duff. Performing the integration gives

$$q_m = \frac{0.57 Q_o}{w} \quad (3)$$

Part of the heat produced by the fire is radiated and convected away at the duff surface, and part flows into the soil. As a perhaps unrealistic starting point, we assumed that all of  $q$  was applied to the soil surface through radiation. This is essentially the condition that would exist if the duff were ignited at the soil surface and burned upward. Such a simulation provides an upper bound for the influence of the duff fire on soil heating.

### Computer Programs

The assumptions just outlined were incorporated into computer codes to simulate soil heating under burning duff. Our earlier computer code was also modified to provide tools for other burn simulations. A disk is enclosed with this report which contains the following programs:

**EXPHEAT.PAS** - A simulation with exponential heating and cooling at the soil surface. The user specifies the maximum value of absorbed radiation at the surface as well as the time constants for heating and cooling.

**DUFFSIM.PAS** - A simulation of burning duff at the soil surface, as outlined in the last section. The user specifies the burn time ( $w$ ), the duff depth, the soil moisture content, and other relevant variables.

**SIMEXPMT.PAS** - The same as DUFFSIM, but with the capability of reading in data from experiments conducted at the Intermountain Fire Laboratory and comparing the simulations with the experimental results.

**BURNSIM.PAS** - A teaching module for fire effects. It allows the user to set values for burn time, duff depth, and soil water content, and then shows a color contour plot of the temperature distribution in the soil resulting from that combination of inputs. There is also an executable version of this program, which can be run by typing BURNSIM.

**SOILHEAT.PAS** - The object of module that initializes and simulates heat and water flow in the soil. This object is used by all of the above listed programs.

Details of the simulation and input variables can be found in our earlier report. We made the following changes to the earlier code. First, we took advantage of some of the features of

object oriented Pascal by creating the object `osoiltemp`. This provides a convenient way of modularizing the main part of the simulation so that it can be initialized, and then stepped through the simulation. One important improvement provided by this approach was that of providing for a simple adaptive time step. We can try a step of a preset size. If the solution is not obtained within a fixed number of iterations a message is sent back to the calling program indicating failure of the step. The calling program then simply reinitializes the object to the values existing at the start of the step, cuts the time step in half, and tries again. This seems to have solved a lot of the problems we had with the earlier program hanging up when soil water content is high.

The previous program read simulation parameters and initial conditions from a file. The present programs give these as constants at the beginning of the program. This requires that the programs be run from the Turbo Pascal IDE environment in order to change parameters, but it makes it very easy to change parameters and compare results. Other minor changes have to do with boundary conditions. The air vapor pressure was set constant and not changed during a simulation. This appears to have little effect on the outcome of the simulation, but adds considerably to the reliability of the program, since it was hard to know the vapor pressures in the fire, and too high a value caused the program to crash. We also eliminated convective heat transfer as a mode of heat exchange with the surface. These versions of the model have only radiative exchange at the surface. Under a fire it was difficult to know what to use for temperatures and exchange coefficients, and the dominant mode of energy transfer is expected to be radiative anyway. Comparison of simulation results showed little effect of convection.

## **Results and Discussion**

The main result of this report is the software that was developed. Some interesting conclusions can be drawn, however, from the comparisons of simulations with measured temperatures under duff fires. The data for these comparisons were provided by Roger Hungerford of the Intermountain Fire Sciences Laboratory. Figures 1..6 compare simulated soil temperatures with measured soil surface temperature. Burn times and times of maximum heat input were adjusted so that the simulated heat pulse approximately coincided with the measured heat pulse.

The interesting feature of these simulations is that only two of the six cases show simulated temperatures exceeding measured temperatures in spite of the fact that we allowed for no heat loss from the fire except to the soil surface. In some cases, much higher heat fluxes would have been needed to make measured and simulated temperatures agree. Since the model was checked against laboratory experiments with measured heat flux densities at the surface, it is hard to see how it could be far wrong. On the other hand, the measurements were straightforward, and should be accurate. I have no explanation of these differences, but feel that they underscore the need for measurement of radiant heat fluxes at the soil surface under fires.

Another interesting feature of the simulations is their relative insensitivity to the amount of fuel, once a critical amount is exceeded. Doubling the duff depth from 10 to 20 cm has a relatively small effect on the depth of lethal temperature penetration, according to the model. Further measurements are needed to check this prediction. Yet another interesting comparison is that of wet soil and dry soil. The effect of fire on a wet soil penetrates deeply, but high

temperatures are restricted to the top few cm. In a dry soil the overall effect is confined to shallower layers, but these layers become much hotter than in the wet soil.

### **Literature**

Campbell, G. S., J. D. Jungbauer, Jr., K. L. Bristow, and W. R. Bidlake. 1992. Simulation of heat and water flow in soil under high temperature (fire) conditions. Final report to the U. S. Forest Service, Intermountain Fire Lab., Missoula MT.

Frandsen, W. H. 1991a. Heat evolved from smoldering peat. *Int. J. Wildland Fire*. 1:197-204.

Frandsen, W. H. 1991b. Smoldering spread rate: a preliminary estimate. paper presented at the 11th Conference on Fire and Forest Meteorology, April 16-19, 1991 at Missoula, MT

Hungerford, R. D., M. G. Harrington, W. H. Frandsen, K. C. Ryan, and G. J. Niehoff. 1991. Influence of fire on factors that affect site productivity. U. S. Forest Service Intermountain Research Station General Technical Report INT-280.

Pafford, D., V. K. Dhir, E. Anderson, and J. Cohen. 1991. Analysis of experimental simulation of ground surface heating during a prescribed burn. *Int. J. Wildland Fire*. 1:125-146.

Steward, F. R., S. Peter, and J. B. Richon. 1990. A method for predicting the depth of lethal heat penetration into mineral soils exposed to fires of various intensities. *Can. J. For. Res.* 20:919-926.

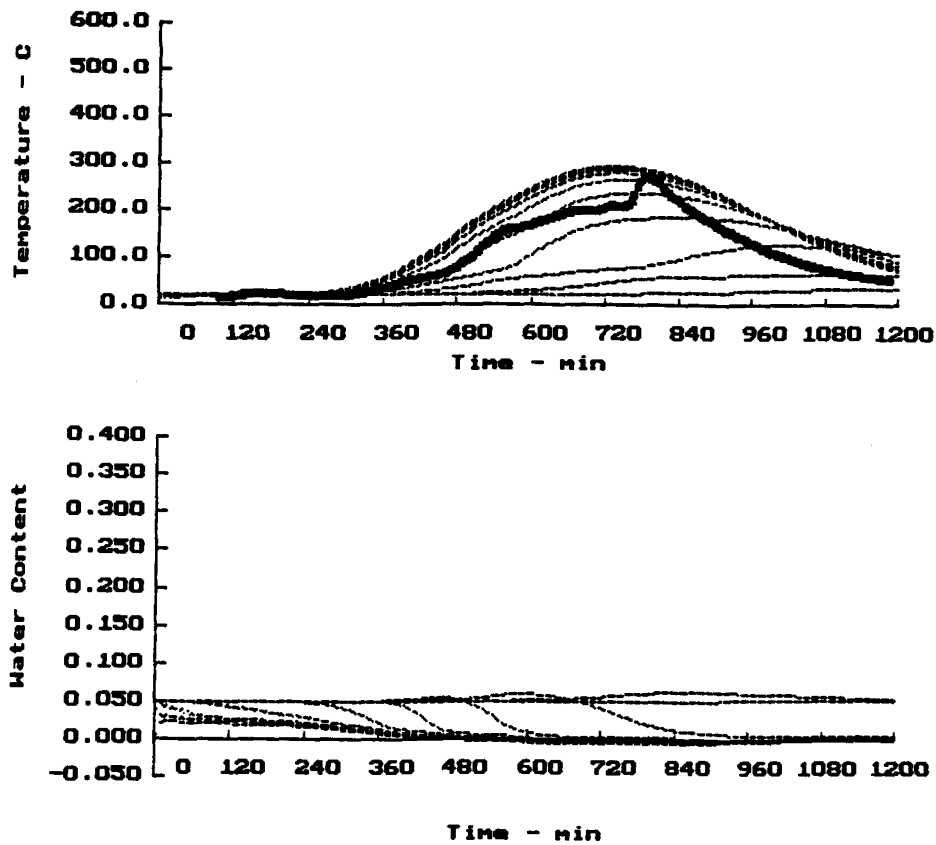


Figure 1. Comparison of simulated soil temperature (dots) and measured surface temperature (heavy line); WAWADRY P1 data. Lower graph is water content.

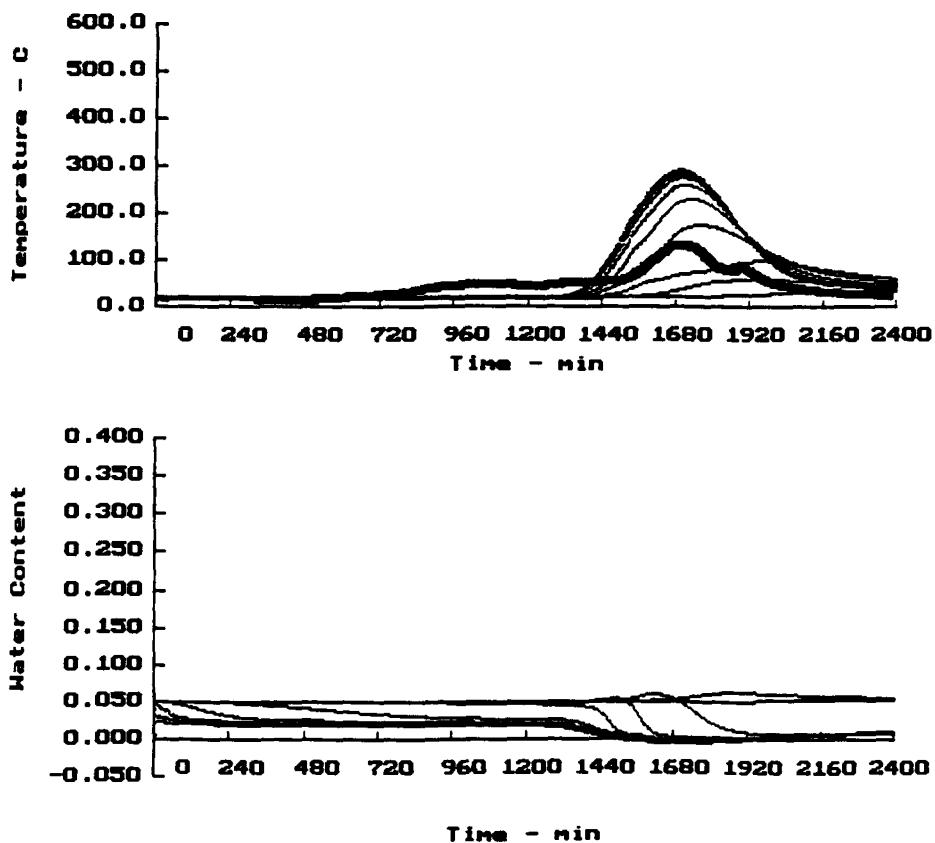


Figure 2. Comparison of simulated soil temperature (dots) and measured surface temperature (heavy line); WAWADRY P2 data. Lower graph is water content.

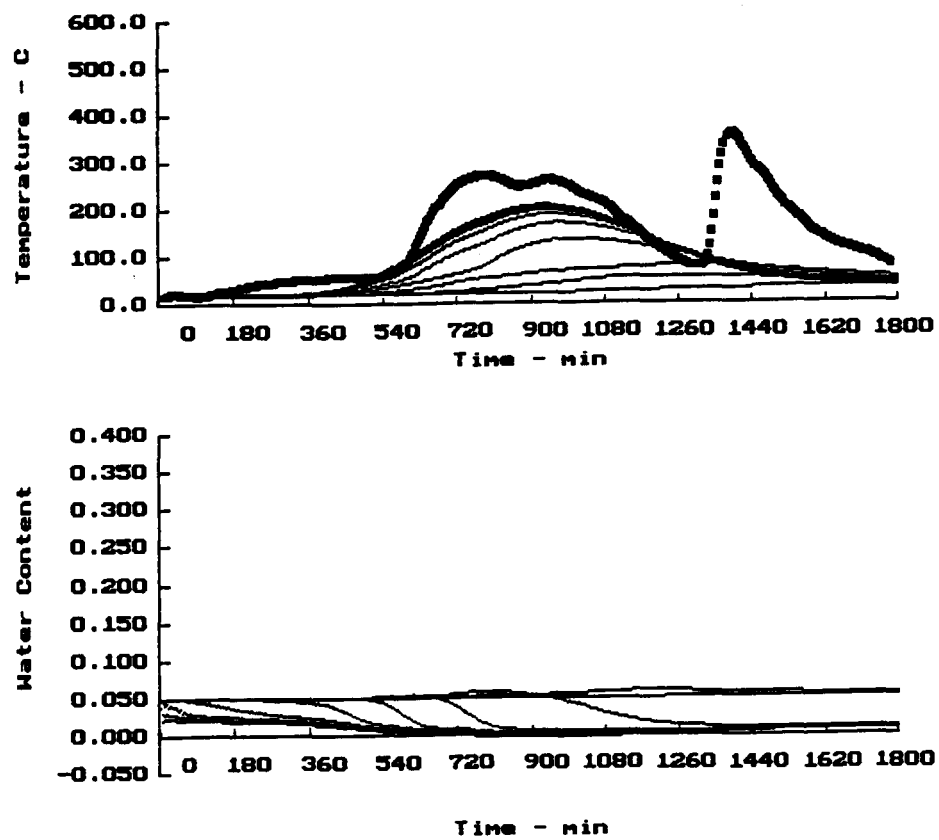


Figure 3. Comparison of simulated soil temperature (dots) and measured surface temperature (heavy line); WAWADRY P3 data. Lower graph is water content.

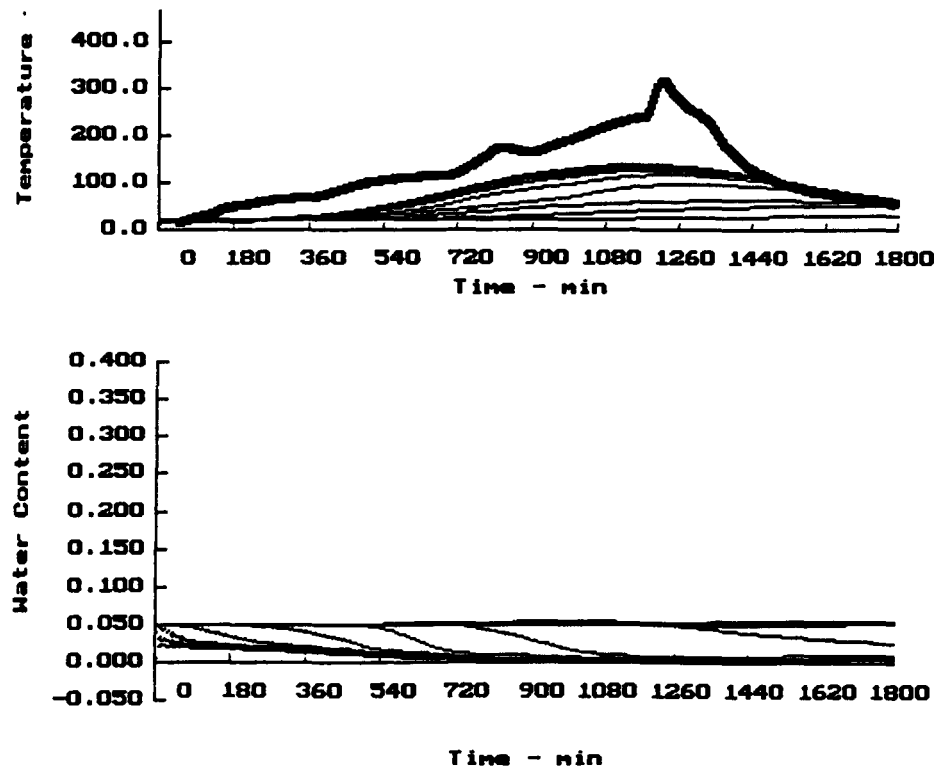


Figure 4. Comparison of simulated soil temperature (dots) and measured surface temperature (heavy line); WAWADRY P4 data. Lower graph is water content.

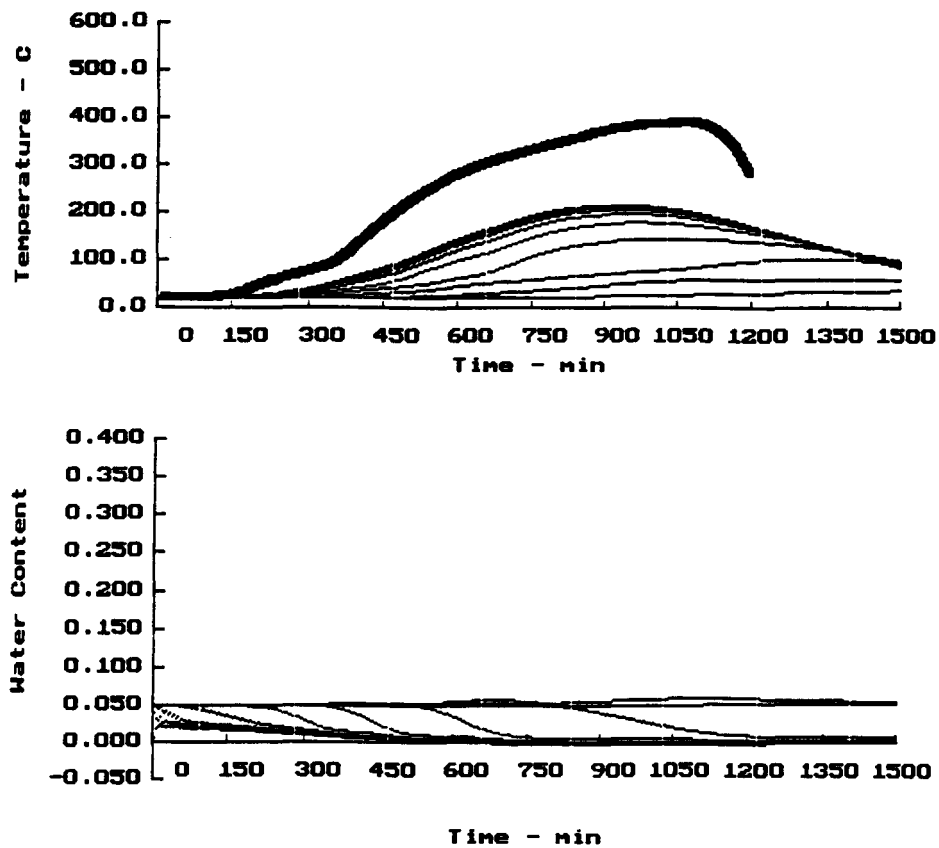


Figure 5. Comparison of simulated soil temperature (dots) and measured surface temperature (heavy line); WLHIGH data. Lower graph is water content.

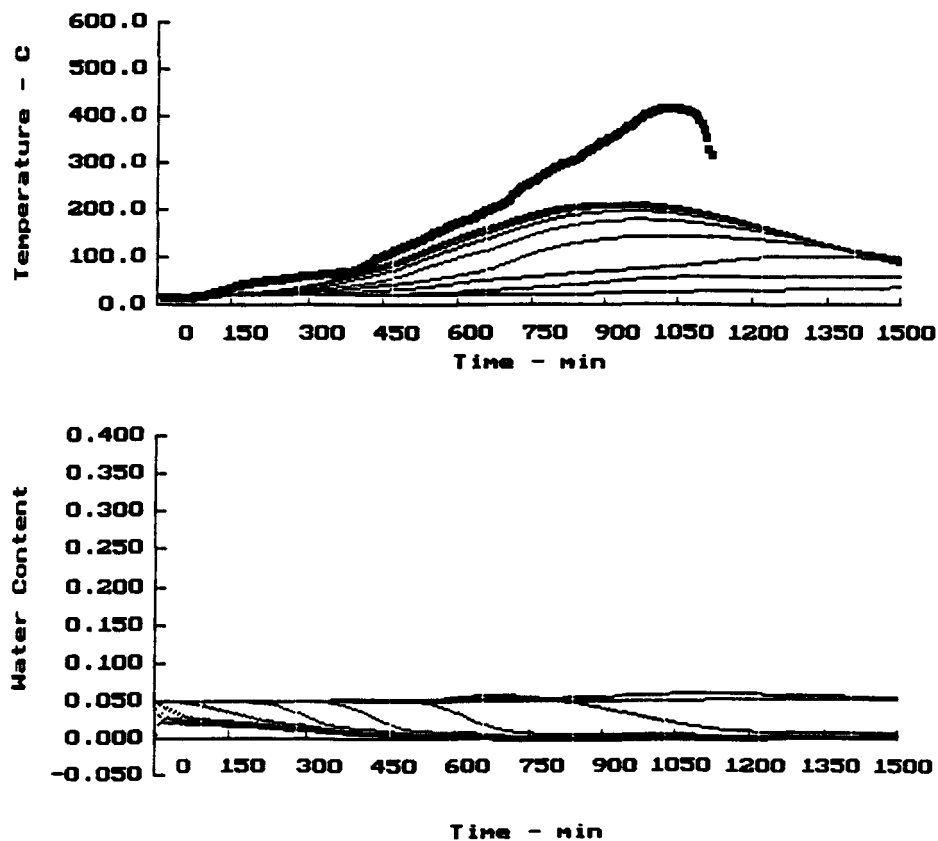


Figure 6. Comparison of simulated soil temperature (dots) and measured surface temperature (heavy line); 12A DUFF data. Lower graph is water content.

## Appendix: Program Listings

Program EXPHEAT;

{ \$R+ }

uses soilheat,scrplt2,crt;

const

```
starttime = 60; {time from simulation start to fire on - min }
burntime = 180; {time from sim start to fire off - min      }
cooltime = 360; {time from start to end of simulation       }
maxRabs = 40000; {maximum radiant heat input to surface     }
heatconst = 60; {time constant for heating, minutes        }
coolconst = 60; {time constant for cooling, minutes         }
bd = 1.22e6;    {soil bulk density - g/m3                   }
pd = 2.65e6;    {soil particle density - g/m3               }
xo = 0.1;       {extrapolated water cont. at -1 J/kg        }
ls = 2.68;      {thermal conductivity of mineral fraction  }
ga = 0.107;     {de Vries shape factor                      }
xwo = 0.126;    {water content for liquid recirculation     }
cop = 2.89;     {power for recirculation function           }
dt = 60;        {time step - s                              }
startwc = 0.2;  {starting soil water content - m3/m3        }
starttemp = 20.0; {starting soil temperatue - C             }
```

var

time,Rabs,Tub,vpub,Rabsub:real;

exh,exc,simdt:real;

w,t,z:rectors;

soiltemp:osoiltemp;

i,j:integer;

success:boolean;

begin

graphsetup(0.1,0.8,0.1,0.4,1);

graphsetup(0.1,0.8,0.5,0.9,2);

scale(0,cooltime,0,600,1);

scale(0,cooltime,-0.05,0.45,2);

axes(0,0,cooltime/10,100,true,1);

axes(0,0,cooltime/10,0.05,true,2);

Rabs:=5.67e-8\*sqr(sqr(starttemp+273.0));

soiltemp.initconsts(bd,pd,ls,ga,xwo,cop,xo);

soiltemp.getdepths(z);

for i:=0 to mplus1 do begin

  w[i]:=startwc; t[i]:=starttemp;

end;

soiltemp.initprofile(w,t);

time:=0;

```

while time < cooltime do begin
  if time<starttime then begin
    exh:=0; exc:=1;
  end;                                     {to start of fire}
  if (time>starttime) and (time<burntime) then {during the burn }
    exh:=(1-exp(-(time-starttime)/HeatConst)); {exp. heating and}
  if (time>burntime) then {cooling during }
    exc:=exp(-(time-burntime)/CoolConst); {and after the burn }
  Rabsub:=maxRabs*exh*exc+Rabs; {set abs. rad. at up. bnd}
  simdt:=dt;
  repeat
    soiltemp.step(Rabsub,dt,success); {simulate a time step }
    if not success then begin {if not successful then}
      simdt:=simdt/2; soiltemp.initprofile(w,t); {cut time step in half }
    end; {and try again }
  until success;
  time:=time+simdt/60;
  soiltemp.getwater(w);
  soiltemp.gettemps(t);
  for i:=1 to mplus1-1 do begin
    pointplt(time,t[i],0,1);
    pointplt(time,w[i],0,2);
  end;
end;
directvideo:=false; write('press Enter to quit '); readln;
end.

```

Program DUFFSIM;

{ \$R+ }

uses soilheat,scrplt2,crt;

const

midburn = 360; {time from sim. start middle of burn - min }  
burntime = 120; {width of burn time at half height - min }  
stoptime = 720; {time from start to end of simulation - min }  
duffheat = 17.5E6; {duff heat content - J/m3 }  
duffdepth = 0.07; {depth of duff layer - m }  
duffdensity = 110; {duff density - kg/m3 }  
bd = 1.22e6; {soil bulk density - g/m3 }  
pd = 2.65e6; {soil particle density - g/m3 }  
xo = 0.1; {extrapolated water cont. at -1 J/kg }  
ls = 2.68; {thermal conductivity of mineral fraction }  
ga = 0.107; {de Vries shape factor }  
xwo = 0.126; {water content for liquid recirculation }  
cop = 2.89; {power for recirculation function }  
dt = 60; {time step - s }  
startwc = 0.10; {starting soil water content - m3/m3 }  
starttemp = 20.0; {starting soil temperatue - C }

var

Qo,qm,time,simdt,Rabs,Rabsub:real;

w,t,z:rectors;

soiltemp:soiltemp;

i,j:integer;

success:boolean;

begin

graphsetup(0.1,0.8,0.1,0.4,1);

graphsetup(0.1,0.8,0.5,0.9,2);

scale(0,stoptime,0,600,1);

scale(0,stoptime,-0.05,0.45,2);

axes(0,0,stoptime/10,100,true,1);

axes(0,0,stoptime/10,0.05,true,2);

Rabs:=5.67e-8\*sqr(sqr(starttemp+273.0));

soiltemp.initconsts(bd,pd,ls,ga,xwo,cop,xo);

soiltemp.getdepths(z);

for i:=0 to mplus1 do begin

w[i]:=startwc; t[i]:=starttemp;

end;

soiltemp.initprofile(w,t);

Qo:=duffheat\*duffdepth\*duffdensity;

qm:=0.57\*Qo/(burntime\*60); {max. rate of heat production; W/m2}

time:=0;

```

while time < stoptime do begin
  Rabsub:=Rabs+qm*exp(-sqr((midburn-time)/burntime));
  simdt:=dt;
  repeat
    soiltemp.step(Rabsub,dt,success);      { simulate a time step }
    if not success then begin              { if not successful then }
      simdt:=simdt/2; soiltemp.initprofile(w,t); { cut time step in half }
    end;                                  { and try again      }
  until success;
  time:=time+simdt/60;
  soiltemp.getwater(w);
  soiltemp.gettemps(t);
  for i:=1 to mplus1-1 do begin
    pointplt(time,t[i],0,1);
    pointplt(time,w[i],0,2);
  end;
end;
end.

```

Program SIMEXPMT;

{ \$R+ }

uses soilheat,scrplt2,crt;

const

```
datafile = 'wawadry.dat';
      {wawadry.dat    wlhigh.prn  12aduff.prn  }
      {P1 P2 P3 P4          }
plotnode = 24; {6  11 17 24    9      6    }
midburn = 1100; {720 1680 900 1100    1000    900  }
burntime = 500; {240 180 300 300    400    400  }
stoptime = 1800; {1200 2400 1800 1800    1500    1500  }
duffdepth = 0.04; {0.08 0.06 0.05 0.04    0.07    0.07  }
duffheat = 18E6; {duff heat content - J/m3          }
duffdensity = 110; {duff density - kg/m3          }
bd = 1.2e6; {soil bulk density - g/m3          }
pd = 2.65e6; {soil particle density - g/m3          }
xo = 0.1; {extrapolated water cont. at -1 J/kg          }
ls = 2.60; {thermal conductivity of mineral fraction          }
ga = 0.107; {de Vries shape factor          }
xwo = 0.126; {water content for liquid recirculation          }
cop = 2.89; {power for recirculation function          }
dt = 240; {time step - s          }
startwc = 0.05; {starting soil water content - m3/m3          }
starttemp = 20.0; {starting soil temperature - C          }
```

var

```
Qo,qm,time,simdt,Rabs,Rabsub,temp:real;
w,t,z:rectors;
soiltemp:osoiltemp;
i,j:integer;
success:boolean;
infile:text;
c:char;
```

begin

```
assign(infile,datafile);
reset(infile);
readln(infile);readln(infile);
graphsetup(0.2,0.8,0.1,0.4,1);
scale(0,stoptime,0,600,1);
axes(0,0,stoptime/10,100,true,1);
labelx('Time - min',1); labely('Temperature - C',1);
graphsetup(0.2,0.8,0.5,0.9,2);
scale(0,stoptime,-0.05,0.45,2);
axes(0,0,stoptime/10,0.05,true,2);
```

```

labelx('Time - min',2); labely('Water Content',2);
while not eof(infile) do begin
  read(infile,time);
  for i:=1 to plotnode do read(infile,temp);
  readln(infile);
  if datafile='wawadry.dat' then pointplt(time*60,temp,1,1)
  else pointplt(time,temp,1,1);
end;
close(infile);
Rabs:=5.67e-8*sqr(sqr(starttemp+273.0));
soiltemp.initconsts(bd,pd,ls,ga,xwo,cop,xo);
soiltemp.getdepths(z);
for i:=0 to mplus1 do begin
  w[i]:=startwc; t[i]:=starttemp;
end;
soiltemp.initprofile(w,t);
Qo:=duffheat*duffdepth*duffdensity;
qm:=0.57*Qo/(burntime*60); {max. rate of heat production; W/m2}
time:=0;
while time < stoptime do begin
  Rabsub:=Rabs+qm*exp(-sqr((midburn-time)/burntime));
  simdt:=dt;
  repeat
    soiltemp.step(Rabsub,dt,success);      {simulate a time step }
  if not success then begin                {if not successful then}
    simdt:=simdt/2; soiltemp.initprofile(w,t); {cut time step in half }
  end;                                     {and try again      }
  until success;
  time:=time+simdt/60;
  soiltemp.getwater(w);
  soiltemp.gettemps(t);
  for i:=1 to mplus1-1 do begin
    pointplt(time,t[i],0,1);
    pointplt(time,w[i],0,2);
  end;
end;
directvideo:=false;
write('dump to HP printer? (y/n) ');
readln(c);
if (c='y') or (c='Y') then hpscreendump;
end.

```

```

Program BURNSIM;
{$R+}
uses soilheat,graph,scrplt2,crt;

const mnumax = 16; {maximum number of menu items}
      stoptime = 720; {time from start to end of simulation - min }
      colorcode:array[0..10] of integer = (8,7,6,12,4,14,15,13,5,9,1);

type filestring = string[20];
      pvectors = array[1..10] of real;
      menuvectors = array[1..mnumax] of string[15];

procedure DisplayText(textfile:filestring);
var infile:text;
    i:integer; c1:char;
    dspstr:string[80];
begin
  clrscr;
  write('Would you like to see the instructions? (y/n) ');readln(c1);
  if (c1='y') or (c1='Y') then begin
    assign(infile,TextFile);
    reset(Infile);
    while not eof(Infile) do begin
      clrscr;
      gotoxy(1,25); write('PRESS CARRIAGE RETURN TO CONTINUE');
      for i:=1 to 23 do begin
        gotoxy(1,i);
        if not eof(Infile) then readln(Infile,Dspstr) else Dspstr:="";
        write(dspstr);
        end;
        readln;
      end;
    end;
    clrscr;
  end;

procedure makegraphs(param:pvectors; menu:menuvectors; n:integer);
const key:array[0..10] of string[7] =
  (' 0- 40',' 40- 80',' 80-120','120-160','160-200','200-240',
   '240-280','280-320','320-360','360-400',' <400');
var i:integer; numberstring:string[12];
begin
  initiated:=0;      {clear graphics screen}
  graphsetup(0.15,0.6,0.05,0.7,1);

```

```

scale(0,stoptime/60,-0.3,0,1);
axes(0,-0.3,stoptime/360,0.1,true,1);
labeled('Depth - m',1);
labelx('Time Hrs.',1);
backcolor(3);
graphsetup(0.65,1,0,1,4);
scale(0,24,0,19,4);
boxgraph(4);
settextstyle(0,0,1); settextjustify(0,2);
for i:=1 to n do begin
  gwrite(1,8-i,0,menu[i],4);
  if i<n-2 then begin
    str(param[i]:4:0,numberstring);
    gwrite(16,8-i,0,numberstring,4);
  end;
end;
gwrite(1,18,0,'Temperature Code - C',4);
for i:=0 to 10 do begin
  gwrite(1,17-0.8*i,0,key[i],4);linecolor(colorcode[i]);
  plt(10,16.9-0.8*i,0,4);plt(19,16.9-0.8*i,1,4);
  plt(10,16.85-0.8*i,0,4);plt(19,16.85-0.8*i,1,4);

end;
end;

```

```

procedure plotgraph(param:pvector);

```

```

const
  duffdensity = 110; {duff density - kg/m3 }
  duffheat = 17.5E6; {duff heat content - J/m3 }
  bd = 1.22e6; {soil bulk density - g/m3 }
  pd = 2.65e6; {soil particle density - g/m3 }
  xo = 0.1; {extrapolated water cont. at -1 J/kg }
  ls = 2.68; {thermal conductivity of mineral fraction }
  ga = 0.107; {de Vries shape factor }
  xwo = 0.126; {water content for liquid recirculation }
  cop = 2.89; {power for recirculation function }
  dt = 90; {time step - s }
  starttemp = 20.0; {starting soil temperatue - C }

```

```

var
  Qo,qm,time,simdt,Rabs,Rabsub,midburn,burntime,duffdepth,startwc:real;
  w,t,z:rvector;
  soiltemp:soiltemp;
  i,j:integer;
  success:boolean;

```

```

begin
  DuffDepth:=param[1]/100; BurnTime:=param[2]; StartWc:=param[3]/100;
  midburn:=stoptime/3;
  Rabs:=5.67e-8*sqr(sqr(starttemp+273.0));
  soiltemp.initconsts(bd,pd,ls,ga,xwo,cop,xo);
  soiltemp.getdepths(z);
  for i:=0 to mplus1 do begin
    w[i]:=startwc; t[i]:=starttemp;
  end;
  soiltemp.initprofile(w,t);
  Qo:=duffheat*duffdepth*duffdensity;
  qm:=0.57*Qo/(burntime*60); {max. rate of heat production; W/m2}
  time:=0;
  while time < stoptime do begin
    Rabsub:=Rabs+qm*exp(-sqr((midburn-time)/burntime));
    simdt:=dt;
    repeat
      soiltemp.step(Rabsub,dt,success);      {simulate a time step }
    if not success then begin                {if not successful then}
      simdt:=simdt/2; soiltemp.initprofile(w,t); {cut time step in half }
    end;                                     {and try again      }
    until success;
    time:=time+simdt/60;
    soiltemp.getwater(w);
    soiltemp.gettemps(t);
    for i:=2 to mplus1 do begin
      j:=trunc((t[i-1]+t[i])/80); if j>10 then j:=10;
      linecolor(colorcode[j]);
      plt(time/60,-z[i-1],0,1); plt(time/60,-z[i],1,1);
    end;
  end;
end;
end;

```

```

procedure getval(var v:real;i:integer);
var instr:string[15];
    code:integer;
    ch:char;
begin
  gwrite(1,1,0,'new value: ',4);
  instr:=""; code:=0;
  repeat
    ch:=readkey;
    if ch in [#46..#57] then begin
      gwrite(12+code,1,0,ch,4);code:=code+1;
      insert(ch,instr,code);
    end;
  until ch in [#0..#32];
end;

```

```

end;
until (ch=#13) or (code>6);
gwrite(1,1,0,' ',4);
if length(instr)>0 then begin
  val(instr,v,code);
  if code=0 then begin
    gwrite(17,8-i,0,' ',4);
    str(v:4:0,instr);
    gwrite(16,8-i,0,instr,4);
  end;
end;
end;

```

```

var burntime,duffdepth,startwc:real;
    np,n,graphdriver,graphmode:integer; ch:char;
    menu:menuvectors;
    param:pvectors;

```

```

begin
  displaytext('burnsim.txt');
  menu[1]:='DuffDepth cm'; menu[2]:='BurnTime min'; menu[3]:='SoilWc %';
  menu[4]:='hit P to Plot'; menu[5]:='hit E to Erase'; menu[6]:='hit Q to Quit';
  burntime:=120; duffdepth:=7; startwc:=20;
  param[2]:=burntime; param[1]:=duffdepth; param[3]:=startwc;
  np:=6;
  makegraphs(param,menu,np);
  n:=1;
  while true do begin
    gwrite(1,8-n,1,menu[n],4);
    ch:=readkey;
    case ch of
      #0: begin
        ch:=readkey;
        if ch=#72 then begin
          gwrite(1,8-n,0,menu[n],4);
          n:=n-1; if n<1 then n:=np;
        end;
        if ch=#80 then begin
          gwrite(1,8-n,0,menu[n],4);
          n:=n+1; if n>np then n:=1;
        end;
      end;
    end;
  #13: case n of
    1: getval(param[1],n);
    2: getval(param[2],n);

```

```

3: getval(param[3],n);
4: getval(param[4],n);
5: getval(param[5],n);
6: getval(param[6],n);
7: getval(param[7],n);
8: begin
    param[8]:=param[8]+1;
    plotgraph(param);
end;
9: begin
    initiated:=0; makegraphs(param,menu,np);
    param[8]:=0;
end;
10: begin
    closegraph;
    exit;
end;
end;
'p','P': begin
    param[8]:=param[8]+1;
    plotgraph(param);
end;
'e','E': begin
    initiated:=0; makegraphs(param,menu,np);
    param[8]:=0;
end;
'q','Q': begin
    closegraph;
    exit;
end;

end;
end;
end.

```

```

{This module determines the temperature and water content of soil that is}
{heated to high temperatures at the surface.                               }
{written Dec. 11, 1989                                                    }
{modified Aug. 1991 for Rabs at surf. and simultaneous solution of h & w }
{modified Sep. 1991 to solve for water potential                         }
{modified Jun. 1994 to be a Turbo Pascal object                          }
{G. S. Campbell                                                            }
{Dept. of Crop and Soil Sciences                                         }
{Washington State University                                             }
{Pullman, WA 99164-6420                                                 }
{phone 509-335-1719                                                      }

```

```

{$R+}
unit soilheat;

```

```

interface

```

```

uses crt;

```

```

const mplus1 = 31; {number of simulation nodes plus 1}
    zmax = 0.3; {depth of lower boundary, m}
    Patm = 92000; {atmospheric pressure at simulation site, Pa}
    Dvo = 2.12e-5; {vapor diffusivity in air, m2/s}
    Tstd = 273.15; {standard temperature, K}
    Po = 101300; {sea level or standard pressure, Pa}
    R = 8.3143; {gas constant, J/mol/K}
    Mw = 0.018; {mole mass of water, kg/mol}
    hc = 20; {surface boundary layer resistance}
    epse = 100; {energy balance error - W/m2}
    epsw = 1e-5; {water mass balance error - kg/(m2 s)}
    dw = 1000; {density of water - kg/m3}
    tor = 0.66; {soil tortuosity - dimensionless}
    maxits = 20; {maximum number of iterations in solution}
    airvp = 1000; {air vapor pressure - Pascals}
    Tair = 20; {starting air temp - C}

```

```

type real = double;
    rsubs = 0..mplus1;
    rvectors = array[rsubs] of real; {Temps. and water contents}
    osoiltemp = object
        conv,vcon,p,z,kv,psat,h,w,wn,t,tn,v,kh,ke,
        kev,cp,s,AirPor,Hvap,dwdp,dhdp,u,enh:rvectors;
        m:integer;
        bd,pd,ls,ga,xwo,cop,xo,xw,xws,xs,ch,wav,tav:real;

```

```

gvol,dC,dv,dCdt,dCdp,dvdp,dvdt,seh,sev,dtm,dJv,dJvdt,dJvdp:real;
ms,ns,tk,tk3:real;
procedure initconsts(bdi,pdi,lsi,gai,xwoi,copi,xoi:real);
procedure initprofile(wi,ti:rvector);
procedure step(Rabs,dt:real; var success:boolean);
procedure getwater(var wi:rvector);
procedure gettemps(var ti:rvector);
procedure getdepths(var zi:rvector);
end;

```

## implementation

```

function pow(x,y:real):real;
begin
  if x<0 then x:=-x;
  if x=0 then pow:=0
  else pow:=exp(y*ln(x));
end;

```

```

function humidity(p,t:real; var dhdp:real):real;
var h,Tk:real;
begin
  Tk:=t+Tstd;
  h:=exp(Mw*p/(R*Tk));
  dhdp:=Mw*h/(R*Tk);
  humidity:=h;
end;

```

```

function watercontent(p, xo:real; var dwdp:real):real;
const lnpo = 13.82; {ln of oven dry water content}
begin
  if p>=0 then p:=-0.001;
  dwdp:=-xo/(lnpo*p);
  watercontent:=xo*(1-ln(-p)/lnpo);
end;

```

```

function vaporpressure(t:real):real; {returns vapor pressure in Pa}
begin
  t:=1-373.15/(t+273.15);
  vaporpressure:=101325*exp(t*(13.3016+t*(-2.042+t*(0.26+t*2.69))));
end;

```

```

function slope(t,p:real):real; {returns slope of vp curve; Pa/C}
var dydt,Tk:real;
begin
  {t in deg. C, p in pascals}

```

```

Tk:=t+Tstd;
t:=1-373.15/Tk; dydt:=373.15/sqr(Tk);
slope:=p*dydt*(13.3015+t*(-4.082+t*(0.78+t*10.76)));
end;

function Hv(t:real):real; {returns latent heat of vaporization in J/kg}
begin
    {t in deg. C}
    Hv:=2.508e6-2670*t;
end;

function Kvap(t,p:real):real; {returns vapor conductivity in kg/(m s Pa)}
var Tk,Dv,stcor:real;
begin
    Tk:=t+Tstd;;
    Dv:=Dvo*(Po/Patm)*pow(Tk/Tstd,1.75);
    stcor:=1-p/Patm;
    if stcor<0.3 then stcor:=0.3;
    Kvap:=Mw*Dv/(R*Tk*stcor);
end;

function tcond(t,xw,xs,ls,ga,xwo,cop,p,s:real; var enh:real):real;
{t in deg. C, xw,xs:vol fract water, solid; p is actual vapor pressure (Pa);
s is slope (Pa/C); returns the thermal conductivity of the soil}
var wf,ka,ks,kw,la,lw,lf,xa,xws,gc,lda,xv,tc:real;
begin
    {xw is vol wc}
    xws:=1-xs; xa:=xws-xw;
    if t<100 then begin
        lw:=0.554+t*(2.24e-3-9.87e-6*t);
        tc:=sqr(sqr((t+273)/303));
    end
    else begin
        lw:=0.68; tc:=2.3;
    end;
    lda:=0.024+t*(7.73e-5-2.6e-8*t);
    if xw<0.01*xwo then wf:=0
    else wf:=1/(1+pow(xw/xwo,-cop*tc));
    la:=lda+wf*Hv(t)*s*Kvap(t,p);
    gc:=1-2*ga; lf:=la+(lw-la)*wf;
    ka:=(2/(1+(la/lf-1)*ga)+1/(1+(la/lf-1)*gc))/3;
    kw:=(2/(1+(lw/lf-1)*ga)+1/(1+(lw/lf-1)*gc))/3;
    ks:=(2/(1+(ls/lf-1)*ga)+1/(1+(ls/lf-1)*gc))/3;
    enh:=(1+2*wf)*ka;
    tc:=(kw*lw*xw+ka*la*xa+ks*ls*xs)/(kw*xw+ka*xa+ks*xs);
    if (tc>0) and (tc<5) then tcond:=tc else tcond:=1;
end;

```

```

procedure osoiltemp.initconsts(bdi,pdi,lsi,gai,xwoi,copi,xoi:real);
const r = 1.6; {multiplier for depth increments}
var i:integer;
begin
  bd:=bdi; pd:=pdi; ls:=lsi; ga:=gai; xwo:=xwoi; cop:=copi; xo:=xoi;
  m:=mplus1-1;
  xs:=bd/pd; xws:=1-xs;
  z[0]:=0; z[1]:=0;
  z[2]:=zmax/m; {z[2]:=zmax/pow(r,mplus1-2);}
  {directvideo := false; writeln(z[2]:8:4);}
  for i:=1 to m do begin
    if i<m then begin
      z[i+2]:=z[i+1]+z[2]; {r*z[i+1];}
      {writeln(z[i+2]:8:4);}
    end;
    v[i]:=0.5*(z[i+1]-z[i-1]);
  end;
  AirPor[0]:=1;
end;

procedure osoiltemp.initprofile(wi,ti:rvector);
var i:integer;
begin
  w:=wi; wn:=wi; t:=ti; tn:=ti;
  for i:=0 to mplus1 do begin
    p[i]:=-exp(13.82*(1-w[i]/xo));
    w[i]:=watercontent(p[i],xo,dwdp[i]);
    h[i]:=humidity(p[i],t[i],dhdp[i]);
    Kev[i]:=0; u[i]:=0; enh[i]:=0;
  end;
end;

procedure osoiltemp.step(Rabs,dt:real; var success:boolean);
var i,its:integer;
begin
  tn[0]:=Tair; its:=0;
  repeat {begin heat and water solutions}
    seh:=0; sev:=0;
    ke[0]:=0 {hc}; {hc taken out so that surf. temp. not needed}
    kev[0]:=6.2e-9*hc; {6.2e-9*hc makes heat and water cond. equal}
    psat[0]:=vaporpressure(tn[0]); h[0]:=airvp/psat[0];
    psat[1]:=vaporpressure(tn[1]); wav:=0.5*(wn[1]+wn[2]);
    s[1]:=slope(tn[1],psat[1]); Hvap[1]:=Hv(tn[1]);
    kh[1]:=tcond(tn[1],wav,xs,ls,ga,xwo,cop,h[1]*psat[1],s[1],enh[1]);
  until success;
end;

```

```

AirPor[1]:=(xws-wav);
kv[1]:=enh[1]*AirPor[1]*tor*Kvap(t[1],psat[1]*h[1]);
for i:=1 to m do begin
  cp[i]:=v[i]*(0.87*bd+4.18e6*wn[i])/dt;
  psat[i+1]:=vaporpressure(tn[i+1]);
  if i<m then begin
    wav:=0.5*(wn[i+1]+wn[i+2]);
    tav:=0.5*(tn[i+1]+tn[i+2])+273;
  end
  else begin
    wav:=wn[mplus1]; tav:=tn[i+1]+273;
  end;
  conv[i]:=0.5*(u[i-1]+u[i])*1200*293/tav;
  vcon[i]:=conv[i]*Mw/(R*1200*293);
  s[i+1]:=slope(tn[i+1],psat[i+1]); Hvap[i+1]:=Hv(tn[i+1]);
  kh[i+1]:=tcond(tn[i+1],wav,xs,ls,ga,xwo,cop,
    h[i+1]*psat[i+1],s[i+1],enh[i+1]);
  ke[i]:=(kh[i])/((z[i+1]-z[i]))+conv[i];
  AirPor[i+1]:=(xws-wav);
  kv[i+1]:=enh[i+1]*AirPor[i+1]*tor*Kvap(t[i+1],psat[i+1]*h[i+1]);
  kev[i]:=(kv[i]+kv[i+1])/(2*(z[i+1]-z[i]))+vcon[i];
  dJv:=kev[i-1]*(psat[i]*h[i]-psat[i-1]*h[i-1])
    -kev[i]*(psat[i+1]*h[i+1]-psat[i]*h[i]);
  dJvdt:=s[i]*h[i]*(kev[i-1]+kev[i]);
  dJvdp:=psat[i]*(kev[i-1]+kev[i])*dhdp[i];
  dC:=ke[i-1]*(tn[i]-tn[i-1])-ke[i]*(tn[i+1]-tn[i])
    +cp[i]*(tn[i]-t[i])-Hvap[i]*dw*v[i]*(wn[i]-w[i])/dt;
  dv:=dJv+dw*v[i]*(wn[i]-w[i])/dt;
  dCdp:=-Hvap[i]*dw*v[i]*dwdp[i]/dt;
  dvdp:=dJvdp+dw*v[i]*dwdp[i]/dt;
  dvdt:=dJvdt;
  dCdt:=ke[i]+ke[i-1]+cp[i];
  if (i=1) then begin
    tk:=tn[1]+273; tk3:=tk*tk*tk;
    dC:=dC-Rabs+5.67e-8*tk*tk3;
    dCdt:=dCdt+4*5.67e-8*tk3;
  end;
  sev:=sev+abs(dv);
  seh:=seh+abs(dC);
  dtn:=(dv*dCdp-dC*dvdp)/(dCdp*dvdt-dCdt*dvdp);
  if dtn<-100 then dtn:=-100;
  tn[i]:=tn[i]-dtn;
  dtn:=(dv-dvdt*dtn)/dvdp; p[i]:=p[i]-dtn;
  if p[i]>0 then p[i]:=(p[i]+dtn)*0.5;
  if p[i]<-1e20 then p[i]:=-1e20;

```